COMPARING METHODS FOR DYNAMIC AIRSPACE CONFIGURATION

Shannon Zelinski, NASA Ames Research Center, Moffett Field, CA
Chok Fung Lai, UC Santa Cruz, Moffett Field, CA

Abstract

This paper compares airspace design solutions for dynamically reconfiguring airspace in response to nominal daily traffic volume fluctuation. Airspace designs from seven algorithmic methods and a representation of current day operations in Kansas City Center were simulated with two times today's demand traffic. A three-configuration scenario was used to represent current day operations. Algorithms used projected unimpeded flight tracks to design initial 24-hour plans to switch between three configurations at predetermined reconfiguration times. At each reconfiguration time, algorithms used updated projected flight tracks to update the subsequent planned configurations. Compared to the baseline, most airspace design methods reduced delay and increased reconfiguration complexity, with similar traffic pattern complexity results. Design updates enabled several methods to as much as half the delay from their original designs. Freeform design methods reduced delay and increased reconfiguration complexity the most.

Nomenclature

c_i = configuration index

d = average total flight delay

n_+ = aircraft gained

n_- = aircraft lost

N = combined aircraft transfer complexity

s_i = simulation iteration index

v_+ = % volume gained

v_- = % volume lost

V = combined volume transfer complexity

α = % flight tracks within 2 miles of boundary

β = avg dist between flow intersections and bounds

γ = num flights with less than 2 min dwell time

Introduction

Airspace management concepts attempt to mitigate required traffic flow management and allow more user preference and traffic flexibility. One form of airspace management is Dynamic Airspace Configuration, that is reconfiguring airspace boundaries to correspond with the prevailing demand traffic and allow more throughput. Today's traffic flexibility is limited in part due to largely static airspace design. New algorithmic methods of airspace design are being developed to allow airspace to change more dynamically and conform to more flexible traffic. A prior comparison of algorithm generated airspace configurations identified strengths and weaknesses of three airspace partitioning algorithms [1,2]. The different algorithms each had different strengths including reduced flight delay, reduced airspace complexity, and more balanced workload among airspace sectors.

However, there were some technical limitations in the previous comparison. Currently, airspace is reconfigured to accommodate traffic demand, and in the future, it will reconfigure more dynamically. However, as a first step, this previous analysis only compared static configurations. Due to airspace design algorithm limitations, the number of sectors was not fixed between compared configurations, making it difficult to assess their relative benefits. These algorithms have since evolved to address identified weaknesses from the comparison as well as address other considerations such as complexity due to reconfiguration and traffic pattern interactions with airspace boundaries. In addition, several other algorithms have since matured to a level sufficient to participate in a comparison study.

This paper presents a next round comparison of newly improved airspace design solutions for dynamically reconfiguring airspace in response to nominal daily traffic volume fluctuation. Improved versions of the three algorithmic airspace design methods from the previous study and five additional methods were compared. Each method designed

three-configuration airspace solutions with projected simulated flight tracks in Kansas City Center at two times today's traffic levels. In addition, each method used updated traffic projections at each reconfiguration time to update subsequent planned reconfigurations. Data analysis compared not only airspace design benefits between different algorithmic methods, but between design updates from a single method to assess the benefit of dynamically updating planned configurations. Additional compared metrics included traffic pattern complexity with respect to airspace boundaries and reconfiguration complexity.

This paper is organized as follows. The Background section provides background on airspace design research and specific design methods compared in this paper. The Method section describes the methods, including experiment setup, scenarios, and metrics. Detailed and summary results are described in the Results section. The paper ends with Conclusions.

Background

Currently, airspace in both the United States and Europe is partitioned into functional blocks that may be combined into fewer large sectors when traffic volume is low or split into more small sectors when traffic volume is high. This is actually a simple and flexible method of adapting to traffic volume fluctuation. However, configuration schedules are generated days in advance based on estimated traffic demand and tactical changes to the configuration schedules are based on managers' personal experience and judgment. Staffing constraints limit the number of sectors that may be open within a particular group of sectors for which each controller is trained. In the United States, these groups of sectors are called areas of specialization (AOS). This also limits feasible combinations of functional airspace blocks within a single area. In addition, overloads may occur in sectors that cannot be further split.

One body of research is developing algorithms to build good realistic functional block-based configuration schedules rather than relying on human judgment [3-6]. Other research focuses on redesigning the airspace boundaries themselves [7-14]. Many of these methods have performed self-assessments and a few have been compared to each other [1,2], but the assessments focused on the designs themselves and not the cost of reconfiguration.

A human-in-the-loop study conducted at NASA Ames tested the feasibility of reconfiguring by moving an airspace boundary on-demand rather than combining and splitting functional airspace blocks [15]. The study found that the reconfiguration operation itself was feasible. However, certain characteristics of boundary designs with respect to traffic pattern geometry and of reconfiguration from one design to the next tended to increase controller workload and decrease acceptability. Many airspace design algorithms have since incorporated these traffic pattern and reconfiguration complexity considerations into their methods.

This paper compares delay reduction benefits, traffic pattern complexity, and reconfiguration complexity of seven airspace design algorithms that use different approaches. Three of these airspace design methods attempt to address the reconfiguration complexity considerations by using elements of the currently used functional airspace blocks in their design. DAU Slices modifies a given configuration by defining five nmi slices of airspace along the shared edge of a sector pair as Dynamic Airspace Units (DAUs) [16] to effectively move airspace boundaries in five nautical mile increments between sector pairs. CombineSplit uses a set of given functional airspace blocks and desired number of sectors as inputs to group the sectors into configurations [17]. FlightLevel [18] starts with the AOS boundaries, and partitions the AOS's vertically by flight level (1,000 ft increments) to achieve the desired number of sectors for a configuration. By contrast, CombineSplit has the option of recombining the base sector units irrespective of AOS boundaries. Inter-AOS reconfiguration is assumed possible in the future with the development of generic airspace tools and procedures that make it easier for individual controllers to work a larger variety of airspace and remain current [19].

Four other airspace design methods compared were more freeform and did not use any elements of the current functional airspace block design. A new Graph-based method [14] partitions a graph representation of the filed flight plan structure and assigns airspace to each graph partition trying to keep intersections and major flow paths away from sector

boundaries. The final three methods compared are improved versions of those compared in previous work [1,2]. These are SectorFlow, CellGeoSect, and Voronoi described below.

SectorFlow [10] clusters flight track points attempting to minimize airspace complexity parameters. Airspace is then assigned to each flight track cluster. An improved version of SectorFlow [20] compared in this paper addresses flow pattern complexity by including several parameters that helped the algorithm keep flow intersections away from sector boundaries and using a gradient search to refine the boundaries after the initial partition.

CellGeoSect is a hybrid of an airspace cell clustering method compared in previous work [11] and an airspace splitting method called GeoSect [21] used to address traffic pattern complexity. The cell clustering method represents the airspace as a tessellation of hexagonal cells and clusters these cells to maximize flow connectivity within and balance flight count between clusters. GeoSect modifies the resulting design by sequentially removing and redefining the boundary between each pair of sectors to avoid geometric constraints such as the length and boundary crossing angles of major flows.

Voronoi [12] represents the airspace using a Voronoi Diagram. A Genetic Algorithm then optimizes the Voronoi Diagram representation to minimize sector overloading. An improved version of this method, used in this comparison, uses a multi-stage process to incorporate different kinds of constraints into the overall optimization [22]. Some constraints added to address traffic pattern complexity include trying to keep intersections and major flow paths away from sector boundaries and minimizing the number of low dwell time flights.

Method

The following subsections describe the experiment setup and metrics used to compare the seven airspace design methods described above.

Experiment Setup

The experiment focused on airspace above 24,000 ft within Kansas City Center (ZKC) in the United States. ZKC presented a good focus center because its airspace design is moderately complex. The entire center shares a common split between low and high altitude airspace of 24,000 ft. In current operations, the airspace above 24,000 ft is routinely reconfigured between combinations of 27 functional airspace blocks (base sectors) within six areas of specialization (AOS). Figure 1 shows the current ZKC sector design, where color indicates AOS and the altitude split between high and super-high sectors is shown with the super-high altitude sector labels.

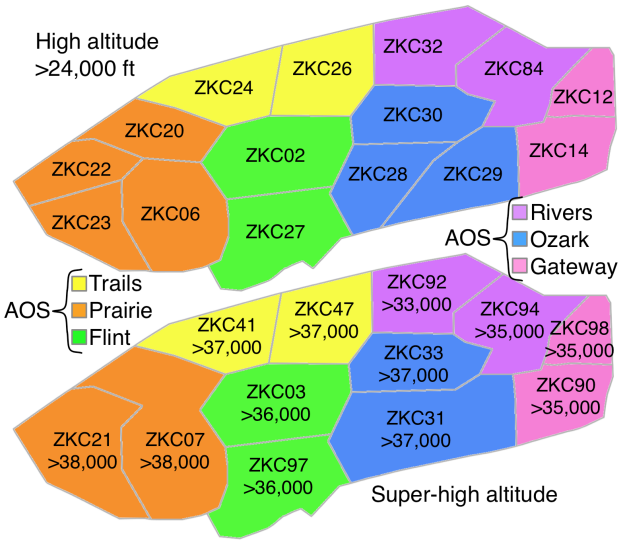

Figure 1. Current ZKC Sector Design

The altitude split between high and super-high altitude sectors ranges between 35,000 and 38,000 ft. Higher altitude over-flights (above 33,000 ft), passing through the center, dominate ZKC's traffic patterns.

Reconfiguration Scenarios

To reduce experiment complexity, all algorithm generated reconfiguration scenarios derived from a three-configuration simplification of historic ZKC airspace operations on 2/8/2007, a nominal day with little weather impact. Lai and Zelinski [23] describes the procedure for processing operational sector combination data and identified three configurations as a reasonable baseline representation for 2/8/2007. All reconfiguration scenarios had the same reconfiguration times and numbers of sectors in each configuration as the Baseline. Given the number of sectors and projected track data for each configuration time period, each algorithm was free to partition the ZKC airspace above 24,000 ft both laterally and vertically. This was a design

improvement over the previous study where algorithms partitioned only laterally within two predefined altitude layers.

Figure 2 diagrams the reconfiguration scenario design. Each box represents a configuration designed for the corresponding time periods. Orange boxes identify active configurations. Black boxes identify planned configurations that were updated before they could be implemented. The reconfiguration times, shown in Central Standard Time, are identical to the Baseline. First, airspace design algorithms use projected flight tracks for the entire day to design an initial three-configuration scenario, labeled as Config 1, 2, and 3, with 6, 24, and 19 sectors above 24,000 ft, respectively. Flight traffic is simulated through Config 1 from 1:45 AM to 8:15 AM, at which point the airspace design algorithm may use updated projected flight tracks to modify the remaining two configurations. These modified configurations are labeled Config 2′ and 3′. The flight traffic simulation continues through Config 2′ from 8:15AM to 9:00PM, at which point the airspace design algorithm may used updated projected flight tracks to modify the last configuration, Config 3′′. The simulation completes through Config 3′′ from 9:00PM to 1:45AM.

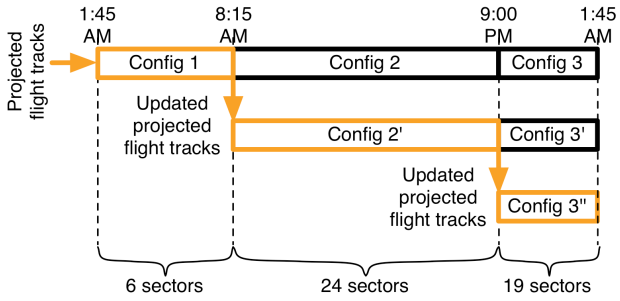

Figure 2. Reconfiguration Scenario Design

Sector capacities for each configuration were assigned using the method presented in Welch et al [24]. This capacity estimation method validated well with respect to historical data using a simple quadratic model based on sector volume and the average flight transit time through the sector during the peak traffic period. The Welch capacity estimation method provided an easily implementable improvement over the previously used method based purely on average flight transit time, which tended to underestimate capacity for large sectors and overestimate capacity for small sectors. The Welch model was used to assign Baseline sector capacities as well. Even though Baseline configurations were not modified at each reconfiguration, updated projected flight tracks required that the sector capacities be updated at each reconfiguration.

Simulation

Simulations were completed using the Airspace Concept Evaluation System (ACES) [25]. ACES has been validated to be a good modeler of en-route trajectories producing similar delay results to real-world operational statistics for good-weather days [26, 27]. ACES modeled gate-to-gate flight operations on airport surfaces and in terminal and en-route airspace. Air traffic control and traffic flow management models controlled flights during these operations to ensure that airspace capacity constraints were not violated. Lower fidelity models were used for ground and airport modeling and higher fidelity models were used for en-route trajectory modeling, which extended from departure meter fixes to arrival meter fixes.

The only constraints imposed in simulation were sector capacities for ZKC airspace above 24,000 ft. Airport and airspace outside of the design scope were unconstrained. It is very difficult to decompose the cause of delays simulated in ACES. Therefore, unconstraining the airports and airspace outside of the design scope ensured that all simulated delay was caused by the ZKC reconfiguration scenario being tested.

ACES simulated flight tracks from the 2/8/2007 flight schedule. Without capacity constraints from airports or neighboring centers or weather impacts, the simulated 2007 traffic produced negligible flight delays. To stress the simulation into producing more delay for analysis, a demand generation tool, AvDemand [28], was used to create a two-times traffic schedule by cloning flights from the 2/8/2007 schedule. AvDemand also time-shifted flights within an hour of the original schedule to reduce unnatural demand peaks.

At the time of this experiment, airspace design algorithms had not been fully integrated into ACES. Therefore, the iterative simulation process in Figure 3 was used to mimic a closed loop simulation of the Figure 2 reconfiguration scenarios.

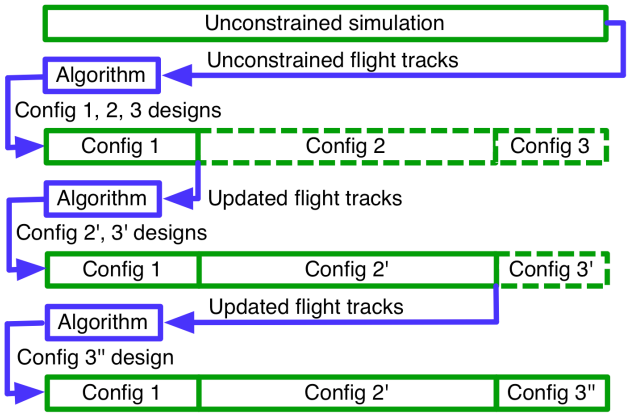

Figure 3. Iterative Simulation Process

Each row of green boxes represents a separate ACES simulation. First, ACES generated unconstrained flight tracks by simulating the 2X 2/8/2007 flight schedule without any capacity constraints. Airspace design algorithms used the unconstrained flight tracks to design three initial configurations. ACES simulated flight traffic subject to Config 1, 2, and 3 constraints through the end of Config 1. Projected flight tracks from this point in the simulation included all traffic modification incurred within the Config 1 time period. Airspace design algorithms used these updated projected flight tracks to design Config 2′ and 3′. ACES simulated flight traffic subject to Config 1, 2′, and 3′ constraints through the end of Config 2′. Airspace design algorithms used updated projected flight tracks to design Config 3′′. Finally, ACES simulated flight traffic subject to Config 1, 2′, and 3′′ all the way to the and of Config 3′′. Even though only the final iteration from Figure 3 represents the full closed loop scenario from Figure 2, all simulations were allowed to complete to analyze the effect each update had on airspace design performance.

Metrics

Metrics were designed to assess the performance of individual configuration designs and reconfigurations between them. Three categories of metrics include, delay, traffic pattern complexity, and reconfiguration complexity. Delay is a user benefit metric. Traffic pattern complexity metrics assess properties of the traffic patterns with respect to airspace boundaries that may affect controller workload. Reconfiguration complexity metrics assess the transition cost from one configuration to the next. Metric details are described below.

Delay

Delay is not only costly to airlines and passengers, but it increases uncertainty by altering flight plans. Reduced delay relative to the current-day baseline simulated with 2x traffic quantifies a user benefit for a set of airspace configurations.

ACES traffic flow management (TFM) monitors sector capacity and projected sector demand with a 6-hour look-ahead time window. TFM issues an entry time restriction to the first flight projected to exceed a particular sector's capacity. Time restrictions may propagate and accumulate as the flight passes through many sectors. Delay may be absorbed en-route with path stretching maneuvers or at the gate as departure delay. At the end of each simulation, the total delay for a flight is the difference between its scheduled and actual gate arrival times.

Because airports and airspace outside ZKC and below 24,000 ft were unconstrained, all flight delay was due to high altitude ZKC airspace capacity constraints. However, it is difficult to quantify the individual delay caused by a particular ZKC configuration. Therefore, average total delays were computed for each three-configuration simulation. Let $d(s_i)$ be the average total delay for ith iteration simulation s_i. There are three simulation iterations, s_1, s_2, and s_3, shown as [Config 1, Config 2, Config 3], [Config 1, Config 2′, Config 3′], and [Config 1, Config 2′, Config 3′′] in Figure 3.

Traffic Pattern Complexity Metrics

Original airspace design algorithms were mostly concerned with minimizing and balancing sector traffic load. However, airspace design must also accommodate traffic pattern geometry to minimize controller cognitive complexity. Most algorithms compared in this study have incorporated some method of aligning the airspace design with traffic patterns to minimize this complexity. The following metrics measure traffic pattern complexities with respect to sector boundaries.

Controllers prefer major flows and their intersections to be well within sector boundaries. To guarantee separation, controllers must be aware of flights not only within a sector, but also just outside the sector. Brinton and Cook [29] show how as-flown

flight paths have a statistically significant lower percentage of flight time within two miles of current sector boundaries (designed to accommodate these paths) than great-circle or wind-optimal paths. The number of aircraft within a threshold distance of a sector boundary was also included in 17 out of 52 original dynamic density metrics found to be significant for measuring airspace complexity [30]. Ideally, flows should stay at least three to five nmi inside the sector boundary to avoid magnifying flight awareness workload of neighboring sectors and to leave room for maneuvering but using a two nmi threshold captures flights that clearly require extra controller attention.

Let $\alpha(s_i c_j)$ be the percentage of flight tracks within two miles of a sector boundary for the jth configuration in the ith simulation iteration. Let α for a particular method be the average $\alpha(s_i c_j)$ of all configurations for all iterations weighted by configuration duration.

Controllers require some time to become familiar with a flight entering the sector before it approaches a major intersection. The more time they have, the more efficiently they may control the flow safely through the intersection. Therefore, intersections should be away from sector boundaries. Brinton and Cook [29] show how there are statistically significantly fewer as-flown flight intersections less than ten miles from a sector boundary than great-circle or wind-optimal path intersections. Jung et al [31] found that increased workload during stable configuration periods correlated to a lower average distance between traffic flow intersections and sector boundaries.

Let $\beta(s_i c_j)$ be the average distance between traffic flow intersections and sector boundaries for the jth configuration in the ith simulation iteration. Let β for a particular method be the average $\beta(s_i c_j)$ of all configurations for all iterations weighted by configuration duration.

Jung et al [31] also found that increased workload during stable configuration periods correlated to the number of flights with short dwell time within a sector. When a flight spends very a small amount time within a sector, controllers often coordinate to directly handoff the flight to the next sector without taking ownership. This causes increased controller workload with no additional service provided to the flight.

Let $\gamma(s_i c_j)$ be the average number of short dwell flights (spending less than two minutes within the a sector) per quarter hour per sector for the jth configuration in the ith simulation iteration.

Reconfiguration Complexity Metrics

It is assumed that reconfiguration incurs an operational cost related to transitioning from one configuration to another. Homola et al [32] showed how new on-demand reconfigurations could be implemented to balance sector traffic load and minimize over-capacity time periods without compromising safety, but at the cost of increasing controller task-load and workload ratings. Lee et al [15] and Jung et al [31] identified percent airspace volume and number of aircraft transferred as the primary contributors to reconfiguration workload for the same study. Percent airspace volume transferred impacts controller situational awareness and number of aircraft transferred impacts controller task-load of handing off aircraft to their new sectors.

The first step for computing reconfiguration metrics between two configurations is to map their sectors to each other. First, sector pairs are mapped in order of decreasing intersection volume. Then, sectors with no intersecting volume are mapping in order of increasing Housdorff distance. Housdorff distance measures how far one sector is spatially shifted from another [33]. Consecutive configurations with different numbers of sectors will have some unmapped sectors assumed to appear or disappear as the sector number increases or decreases, respectively. Let $v_+(k_1,k_2)$ and $v_-(k_1,k_2)$ be the volume gained and lost for sector pair (k_1,k_2) given as

$$v_+(k_1,k_2) = v(k_1) - v_\cap(k_1,k_2)$$
$$v_-(k_1,k_2) = v(k_2) - v_\cap(k_1,k_2)$$

where k_1 is the old sector, k_2 is the new sector, and $v_\cap(k_1,k_2)$ is the shared volume between k_1 and k_2 seen in figure 4.

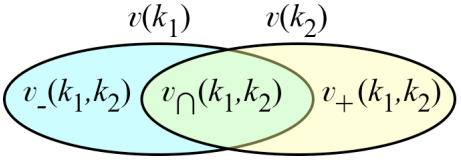

Figure 4. Volume Gained and Lost

Let $v_+(k_1,k_2)$ and $v_-(k_1,k_2)$ be the percent volume gained and lost with respect to the old sector volume.

$$v_+(k_1,k_2) = \upsilon_+(k_1,k_2) / \upsilon(k_1)$$
$$v_-(k_1,k_2) = \upsilon_-(k_1,k_2) / \upsilon(k_1)$$

For unmapped appearing sectors, $\upsilon_+(-,k_2)=\upsilon(k_2)$ and $v_+(-,k_2)=100\%$. For unmapped disappearing sectors, $\upsilon_-(k_1,-)= \upsilon(k_1)$ and $v_-(k_1,-)=100\%$.

Let $V(k_1,k_2)$ be a weighted combined volume transfer complexity given by

$$V(k_1,k_2) = w_{v+} v_+(k_1,k_2) + w_{v-} v_-(k_1,k_2)$$

where w_{v+} and w_{v-} are weighting factors.

Operational reconfigurations can be completed in roughly five minutes [15]. Therefore, the numbers of aircraft gained ($n_+(k_1,k_2,t)$) and lost ($n_-(k_1,k_2,t)$) between k_1 and k_2 are the numbers of unique aircraft flying in $\upsilon_+(k_1,k_2)$ and $\upsilon_-(k_1,k_2)$, respectively, during the five minutes preceding reconfiguration time t. Let $N(k_1,k_2,t)$ be a weighted combined aircraft transfer complexity given by

$$N(k_1,k_2,t) = w_{n+} n_+(k_1,k_2,t) + w_{n-} n_-(k_1,k_2,t)$$

where w_{n+} and w_{n-} are weighting factors.

Lai and Zelinski [23] found that in current reconfiguration operations, there is an average of two aircraft gained and two aircraft lost. Clustering operational reconfigurations into the simplified three-configuration set used in this study altered these values because the clustered reconfiguration times were no longer coordinated with the traffic. The aircraft gained and lost metrics are very sensitive to reconfiguration time due to traffic fluctuation. In operation, managers would be free to implement a reconfiguration any time within a range to minimize the aircraft transfer complexity. Therefore, $N(k_1,k_2,t)$ was calculated for t ranging from thirty minutes before to thirty minutes after the reconfiguration design time in five-minutes increments. It was assumed that the reconfiguration could occur within any of these five-minute intervals, but the entire reconfiguration must be completed within the same interval. This assumption made component-based airspace design methods such as Baseline, CombineSplit, DAU Slices, and FlightLevel, that could reconfigure incrementally, more comparable to the freeform design methods that may require the reconfiguration to occur all at once. It was also assumed that managers would choose the time that minimized the maximum aircraft transition workload.

Results

The following subsections present results for the Baseline and designs from seven airspace design methods.

Delay (d)

Delay measured the benefits of each airspace design method from the user prospective. Lower delay demonstrated user benefits. Figure 5 shows the average total delay for each of the three-configuration simulation iterations shown in Figure 3. Five of the methods produced lower delay than Baseline with their original designs in s_1. After the first design update, all but CombineSplit reduced delay below Baseline. The most significant delay reduction is due to the first design update, between s_1 and s_2. Very little if any delay reduction is achieved with the second design update, between s_2 and s_3. CellGeoSect showed the most user benefit, reducing the Baseline delay by more than two thirds. In general, freeform methods produced lower delays than methods using Baseline components.

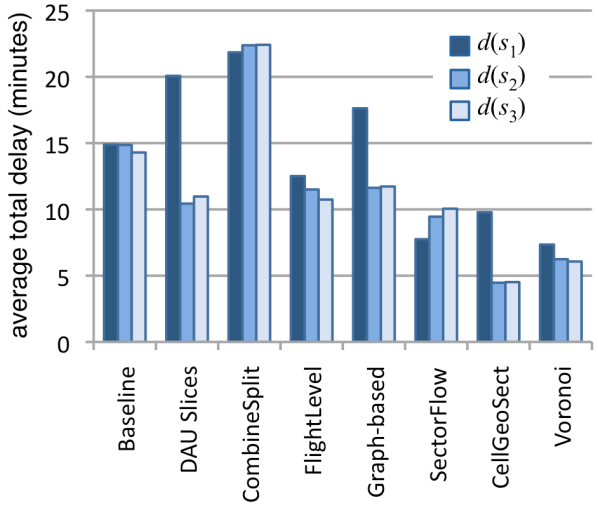

Figure 5. Average Total Delay

Traffic Pattern Complexity

Flight Track Percentage Close to Boundary (α)

The percentage of flights tracks within two nmi of a sector boundary (α) was computed for each configuration and iteration. Figure 6 shows average α results.

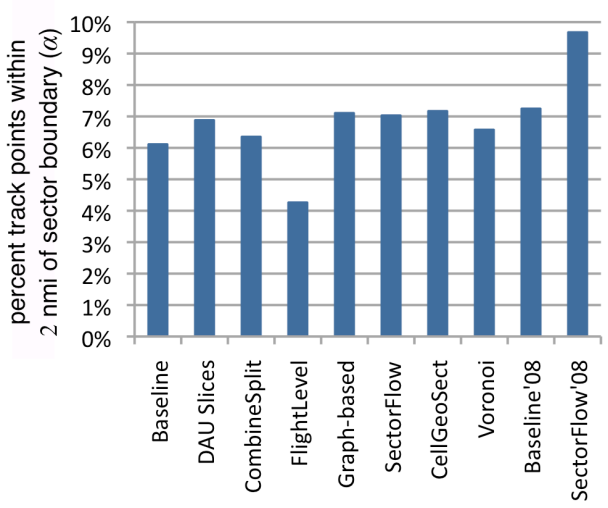

Figure 6. Average α Results

Briton and Cook [29] calculated distributions of percent flight time within two nmi of nation-wide sector boundaries for as-flown tracks with respect to the current airspace design and with respect to their 2008 version of SectorFlow airspace design method. The Baseline'08 and SectorFlow'08 values in Figure 6 were calculated by combining the current airspace design and 2008 SectorFlow design distributions from Figure 6 in [29]. Even though the ZKC Baseline has lower α than the nation-wide Baseline'08, the improvement from SectorFlow'08 to SectorFlow is clear. All other methods besides FlightLevel are between Baseline and Baseline'08. The improvement of FlightLevel over Baseline is because the method uses existing AOS footprints, which have larger lateral area than most individual Baseline sectors. The overall results indicate that all methods compared in this study do a sufficient job of keeping major flows away from sector boundaries.

Flow Intersection Boundary Proximity (β)

Figure 7 shows β for each airspace design method. The β values for Baseline'08 and SectorFlow'08 were calculated from Figure 8 in [28] by multiplying the x and y axis for each column and summing. Baseline and Baseline'08 are very similar and all methods except SectorFlow'08 have higher β than both Baselines. The SectorFlow improvement over SectorFlow'08 moved flow intersections 2.4 miles farther from sector boundaries on average, which is approximately 20 seconds of flight time. Just as with α, FlightLevel has the best β values due to it's use of AOS footprints.

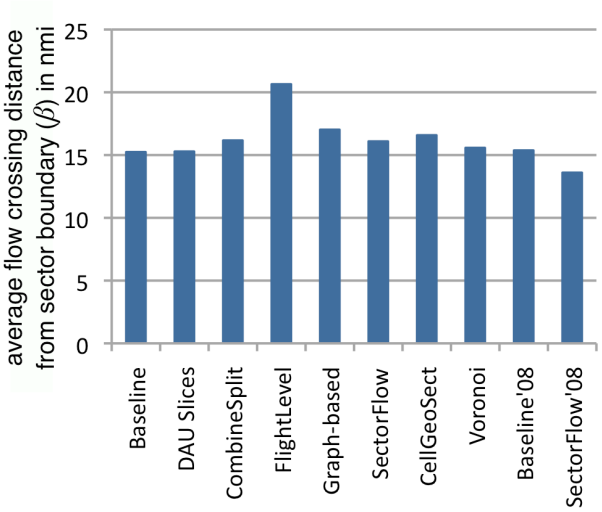

Figure 7. Average β Results

Number of Short Dwell Flights (γ)

All of the design methods indirectly try to maximize average flight dwell time through each sector because it is directly related to maximizing sector capacity. However, only Voronoi explicitly tried to minimize the number of short dwell flights. Figure 8 shows γ averages and quartiles for each airspace design method. Voronoi and CombineSplit are the methods with the most similar or lower γ values than Baseline. CombineSplit is very similar to Baseline because it uses the same base airspace volumes. Voronoi is similar or better than Baseline because it is the only method that explicitly tried to minimize the number of short dwell flights.

FlightLevel sticks out with γ values that are consistently more than twice that of Baseline. Due to FlightLevel vertical partitioning, as the number of sectors increases, sector vertical range decreases. The difference in γ between FlightLevel and Baseline is entirely due to climbing or descending flights passing

through sectors spanning only two or three flight levels. This did not negatively affect FlightLevel's α or β because distances for these metrics are measured relative to lateral boundaries only.

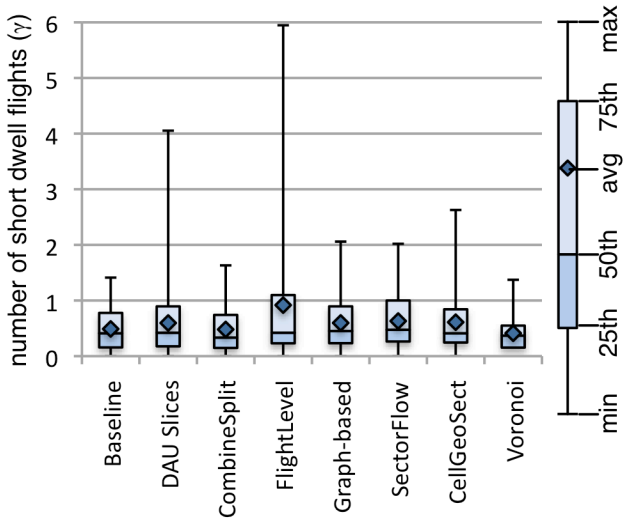

Figure 8. Averages and Quartiles for γ Results

DAU Slices also had a high maximum γ, which is surprising because the method made minimal modifications to Baseline designs. However, these modifications did not explicitly consider traffic pattern complexity and sometimes resulted in sharp boundary angles or panhandles. DAU Slices results demonstrate how small changes can have a large impact.

Reconfiguration Complexity

Volume Transition Complexity (V)

Figures 9 and 10 show averages and quartiles of all V between pairs of mapped sectors for the first and second reconfiguration, respectively. V was calculated using $w_{V+} = w_{V-} = 0.5$.

As seen in Figure 9, only FlightLevel has lower V than Baseline. All other methods have slightly higher V than Baseline and freeform methods have higher V than those using Baseline components. This trend is exaggerated in the second reconfiguration seen in Figure 10. Most methods produce more varied results than Baseline. High maximums were caused by mapped sector pairs with little or no overlapping volume. V tended to be larger in the second reconfiguration than the first for two reasons. V is a weighted percentage of the sector size prior to reconfiguration and c_2 sector sizes were the smallest. Also, the second configuration reduced the number of sectors causing volume gained to dominate V, whereas volume lost dominated V in the first reconfiguration. There was no limit on how much volume a sector could gain but the most volume a sector could lose was 100%.

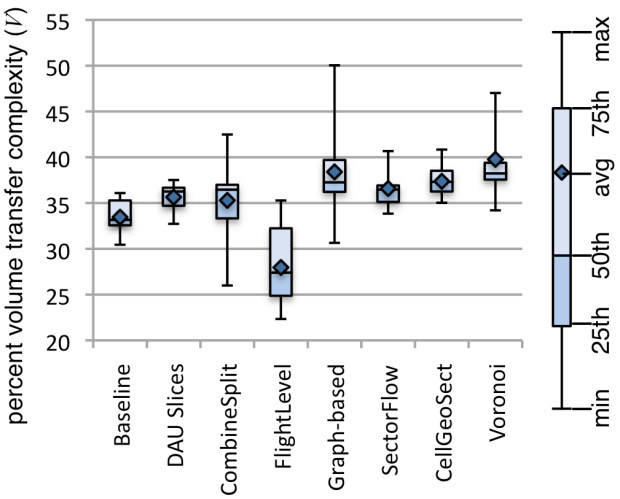

Figure 9. V Results for First Reconfiguration

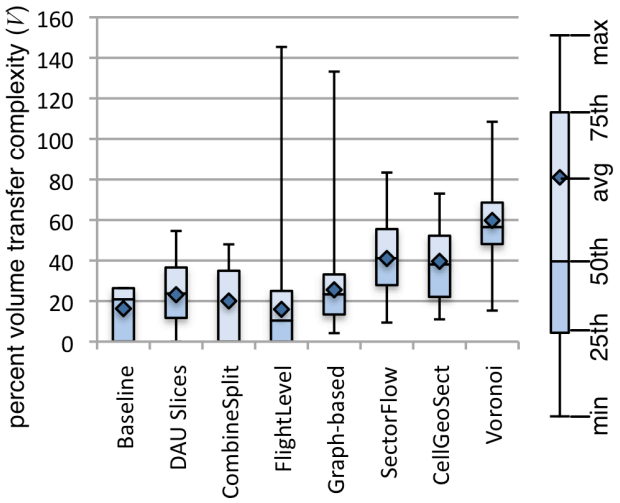

Figure 10. V Results for Second Reconfiguration

Aircraft Transition Complexity (*N*)

For each simulation iteration and reconfiguration, the reconfiguration time *t* used for *N* calculation was the time that minimized the maximum *N* value. The *N* calculations used w_{n+} =1 and w_{n-} =0.2 assuming that accepting and becoming familiar with new aircraft requires much more workload than simply handing-off aircraft to another sector. Figures 11 and 12 display *N* averages and quartiles for the first and second reconfiguration, respectively.

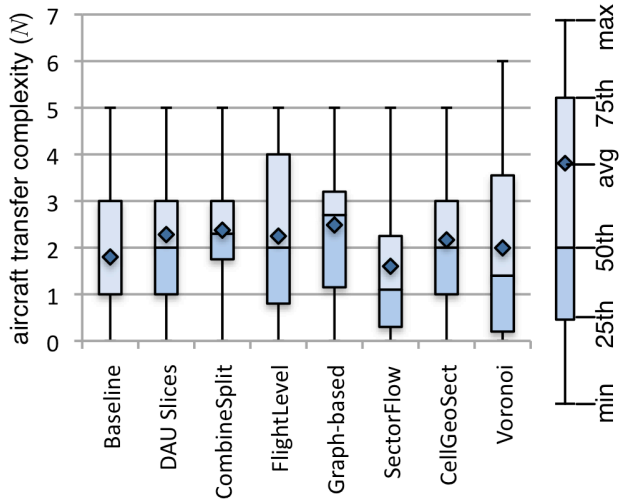

Figure 11. *N* Results for First Reconfiguration

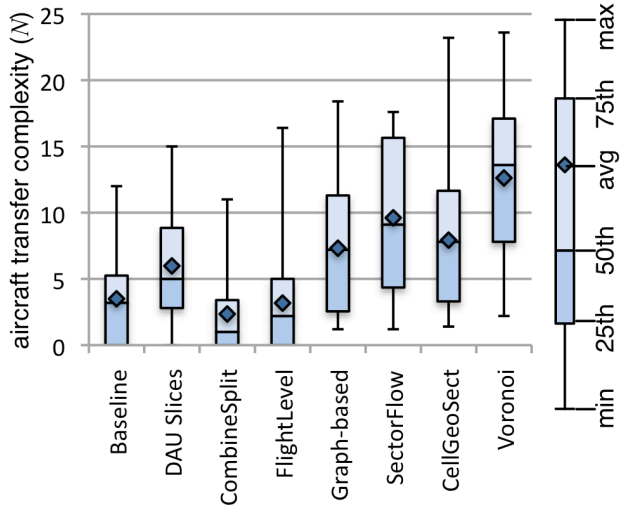

Figure 12. *N* Results for Second Reconfiguration

All methods have very similar *N* values in the first reconfiguration seen in Figure 11. All averages are very close to two and only Voronoi has a maximum greater than the rest, by just one aircraft transfer. The second reconfiguration has much more variation between methods. Figure 12 shows generally higher *N* values for freeform methods than those using Baseline components. Only CombineSplit has consistently lower second reconfiguration *N* than Baseline. By contrast, SectorFlow and Voronoi produce two to three times higher *N* than Baseline.

Results Summary and Discussion

Average airspace design performance is summarized as a percent increase or decrease from Baseline in Table 1. Yellow cells with values close to zero are similar to Baseline. Red cells with negative values are worse and green cells with positive values are better than Baseline. Darker shaded red and green cells have increasingly worse or better results, respectively. Delay performance, *d*, is based on $d(s_3)$ from Figure 5. Traffic pattern complexity performances are based on values from Figure 6 for α, Figure 7 for β, and averages from Figure 8 for γ. Reconfiguration complexity performances are based on averages of the averages from Figures 9 and 10 for *V* and from Figures 11 and 12 for *N*.

Table 1. Airspace Design Performance Summary

	DAU Slices	CombineSplit	FlightLevel	Graph-based	SectorFlow	CellGeoSect	Voronoi
d	23	-57	25	18	30	68	58
α	-13	-4	30	-16	-15	-17	-8
β	0	6	35	12	6	9	2
γ	-22	2	-88	-22	-29	-25	16
V	-18	-11	12	-29	-56	-55	-100
N	-56	11	-2	-85	-112	-90	-176

Worse Similar Better

The ultimate goal of each algorithm was to provide user benefits by reconfiguring airspace. The results show positive delay reduction benefits (*d*) in all but one algorithm, achieving the algorithms' goal.

A few algorithms do better than others at minimizing traffic pattern complexity (α, β, γ), but all do a fairly decent job. In general, algorithms that aggressively change the airspace show more delay reduction benefits but at higher reconfiguration costs (V, N). The reconfiguration costs are expected and are acceptable as long as they are manageable.

The three methods using Baseline elements performed very differently. DAU Slices achieved modest benefits with modest negative effects to traffic pattern and reconfiguration cost. This was expected as DAU Slices is the most conservative method, designed to allow small changes to existing airspace design at high reconfiguration frequency.

CombineSplit was the only method to worsen d. All other CombineSplit metrics are similar to Baseline. This method was designed for a more tactical application, suggesting configurations every 15 minutes over a two-hour horizon. CombineSplit actually decreased delay in a study comparing DAC benefits when applied to a more tactical two-hour weather rerouting scenario when number of sectors remained the same [34].

FlightLevel is the most unique case with widely varying results. It achieved modest d improvement without negatively affecting reconfiguration complexity. However, the traffic pattern results suggest that more research is needed to determine if FlightLevel configurations are feasible. FlightLevel significantly improved α and β metrics because these metrics did not consider vertical boundaries. The significantly worsened γ due to flights climbing or descending through sectors only a few flight levels thick may not be acceptable.

Most recent freeform algorithm development has focused on improving traffic pattern complexity. The improvement of SectorFlow from SectorFlow'08 in this area was demonstrated in the Traffic Pattern Complexity subsection. The most aggressive freeform methods (Voronoi, SectorFlow, and CellGeoSect) produced the greatest delay reduction benefit, but they also significantly increased reconfiguration complexity. Voronoi was the only method to reduce delay relative to Baseline without negatively affecting traffic pattern complexity, making this the most attractive method if the reconfiguration complexity increase is manageable. Reconfiguration complexity thresholds when using DataComm-based controller tools such as those used in [15] have yet to be determined. With the right controller tools and further algorithm refinement to reduce reconfiguration complexity, achieving the higher benefits of these more aggressive methods may be feasible.

Conclusions

A fast-time simulation study compared the performance of solutions from new airspace design methods to a representation of current day dynamic airspace operations. Three categories of metrics compared delay reduction benefits, traffic pattern complexity, and reconfiguration complexity. Most methods achieved benefits by decreasing delay, which was augmented by allowing strategic airspace design updates. Most methods also did a reasonable job of keeping traffic pattern complexity low. Methods using design elements from Baseline had more modest benefits and reconfiguration complexity. Freeform airspace design methods achieved the highest benefits and highest increase in reconfiguration complexity. Future research is needed to determine if high reconfiguration complexity is acceptable given the right controller tools. Airspace design methods may also further refine their algorithms to minimize reconfiguration complexity.

Acknowledgements

The authors thank all algorithm developers who submitted reconfiguration scenarios for this study.

References

[1] Zelinski, S., 2009, A Comparison of Algorithm Generated Sectorizations. In 8th USA/Europe ATM R&D Seminar, Napa Valley, California.

[2] Zelinski, S., 2010, A Comparison of Algorithm Generated Sectorizations. In Air traffic Control Quarterly, Vol. 18, No. 3, pp. 279-301.

[3] Verlhac, C. and S. Manchon, 2001, Optimization Of Opening Schemes. In 4th USA/Europe ATM R&D Seminar, Santa Fe, New Mexico.

[4] Bichot, C. E. and N. Durand, 2007, A Tool to Design Functional Airspace Blocks. In 7th USA/Europe ATM R&D Seminar, Barcelona, Spain.

[5] Gianazza, D., C. Allignol, and N. Saporito, 2009, An Efficient Airspace Configuration Forecast. In 8th USA/Europe ATM R&D Seminar, Napa Valley, California.

[6] Bloem, M., P. Gupta, and P. Kopardekar, 2009, Algorithms for Combining Airspace Sectors. In Air traffic Control Quarterly, Vol. 17, No. 3, pp. 245-268.

[7] Trandac, H., P. Baptiste, V. and Duong, 2003, Airspace Sectorization by Constraint Programming. In Proceedings de la 1re confrence en Recherche Informatique Vietnam & Francophone (RIVF).

[8] Ehrmanntraut, R. and S. McMillan, 2007, Airspace Design Process for DYnamic Sectorisation. In 26th Digital Avionics System Conference, Dallas, Texas.

[9] Conker, R.S., D.A. Moch-Mooney, W.P. Niedringhous, B.T. and Simmons, 2007, New Process for "Clean Sheet" Airspace Design and Evaluation. In 7th USA/Europe ATM R&D Seminar, Barcelona, Spain.

[10] Brinton, C. and S. Pledgie, 2008, Airspace Partitioning Using Flight Clustering and Computational Geometry, In 27th Digital Avionics System Conference, October 26-30, St. Paul, Minnesota.

[11] Drew, M., 2008, Analysis of an Optimal Sector Design Method. In 27th Digital Avionics System Conference, St. Paul, Minnesota.

[12] Xue, M., 2008, Airspace Sector Redesign Based on Voronoi Diagrams. In AIAA Guidance, Navigation and Control Conference and Exhibit, August, Honolulu, Hawaii. AIAA-2008-7223.

[13] Tien, S. and R. Hoffman, 2009, Optimizing Airspace Sectors for Varying Demand Patterns Using Multi-Controller Staffing. In 8th USA/Europe ATM R&D Seminar, Napa Valley, California.

[14] Li, J., T. Wang, and I. Hwang, 2009, A Spectral Clustering Based Algorithm for Dynamic Airspace Configuration, In 9th AIAA Aviation Technology, Integration and Operations Conference, September 21-23, Hilton Head, South Carolina, AIAA-2009-7056.

[15] Lee, P.U., T. Prevot, J. Homola, H. Lee, Kessell, A., C. Brasil, and N. Smith, 2010, Impact of Airspace Reconfiguration on Controller Workload and Task Performance, In 3rd Intl. Conference on Applied Human Factors and Ergonomics, Miami, Florida.

[16] Dynamic Airspace Configuration Benefits. Report submitted by M. Rodgers et al, under NASA Contract NNA07BB31C, Washington, DC, August 2008.

[17] Bloem, M. and P. Gupta, 2010, Configuring Airspace Sectors with Approximate Dynamic Programming, In 27th Congress of the International Council of the Aeronautical Sciences, September 19-24, Nice, France.

[18] Leiden, K., S. Peters, S. and Quesada, 2009, Flight Level-based Dynamic Airspace Configuration, In 9th AIAA Aviation Technology, Integration and Operations Conference, September 21-23, Hilton Head, South Carolina, AIAA-2009-7104.

[19] Mogford, R., 2010, Generic Airspace Concepts and Research, In 10th AIAA Aviation Technology, Integration and Operations Conference, September 13-15, Fort Worth, Texas, AIAA-2010-9159.

[20] Brinton, C., K. Leiden, and J. Hinkey, 2009, Airspace Sectorization by Dynamic Density, In 9th AIAA Aviation Technology, Integration and Operations Conference, September 21-23, Hilton Head, South Carolina, AIAA-2009-7102.

[21] Sabhnani, G., A. Yousefi, and J. Mitchell, 2010, Flow Conforming Operational Airspace Sector Design, In 10th AIAA Aviation Technology, Integration and Operations Conference, September 13-15, Fort Worth, Texas, AIAA-2010-9377.

[22] Xue, M., 2010, Three Dimensional Sector Design with Optimal Number of Sectors, In AIAA Guidance, Navigation, and Control Conference and Exhibit, August 2-5, Toronto, Ontario Canada.

[23] Lai, C.F. and S. Zelinski, 2010, Operational Dynamic Configuration Analysis. In 29th Digital Avionics System Conference, October 3-7, Salt Lake City, Utah.

[24] Welch, J., 2007, Macroscopic Workload Model for Estimating EnRoute Sector Capacity. In 7th USA/Europe ATM R&D Seminar, Barcelona, Spain.

[25] Meyn, L., R. Windhorst, K. Roth, D. VanDrei, G. Kubat, V. Manikonda, S. Roney, G. Hunter, A. Huang, and G. Couluris, 2006, Build 4 of the Airspace Concept Evaluation System. In AIAA

Modeling and Simulation Technologies Conference an Exhibit, Keystone, Colorado. AIAA- 2006-6110.

[26] Zelinski, S., 2005, Validation of the Airspace Concept Evaluation System Using Real World Data. In AIAA Modeling and Simulation Technologies Conference and Exhibit, San Francisco, California.

[27] Zelinski, S. and L. Meyn, 2006, Validation of the Airspace Concept Evaluation System for Multiple Weather Days. In AIAA Modeling and Simulation Technologies Conference an Exhibit, Keystone, Colorado.

[28] Huang, A. and D. Schleicher, 2008, Futuristic US Flight Demand Generation Approach Incorporating Fleet Mix Assumptions. In AIAA Modeling and Simulation Technologies Conference an Exhibit, 18 - 21 August 2008, Honolulu, Hawaii. AIAA 2008-6678.

[29] Brinton, C. and L. Cook, 2008, Analysis of Current Airspace Operations and Implications for Dynamic Airspace Configuration. In AIAA Modeling and Simulation Technologies Conference and Exhibit, August 18-21, Honolulu, Hawaii, AIAA-2008-7224.

[30] Kopardekar, P., 2007, Airspace Complexity Measurement: An Air Traffic Control Simulation Analysis. In 7th USA/Europe ATM R&D Seminar, Barcelona, Spain.

[31] Jung, J., P. Lee, A. Kessell, J. Homola, and S. Zelinski, 2010, Effect of Dynamic Sector Boundary Changes on Air Traffic Controllers. In AIAA Guidance, Navigation, and Control Conference and Exhibit, August 2-5, Toronto, Ontario Canada, AIAA-2010-8289.

[32] Homola, J., P. Lee, T. Prevot, H. Lee, A. Kessell, C. Brasil, N. and Smith, 2010, A Human-in-the-Loop Exploration of the Dynamic Airspace Configuration Concept. In AIAA Guidance, Navigation and Control Conference and Exhibit,

[33] Yousefi, A., R. Hoffman, M. Lowther, B. Khorrami, H. and Hackney, 2009, Trigger Metrics for Dynamic Airspace Configuration, In 9th AIAA Aviation Technology, Integration and Operations Conference, September 21-23, Hilton Head, South Carolina, AIAA-2009-7103.

[34] Jung, J., C.F. Lai, and S. Zelinski, 2010, Analysis of Regional Airspace Reconfigurations in Presence of Convective Weather, NASA Milestone report for AS.2.3.05.